```
import pandas as pd
```

```
df = pd.read_csv('/content/QVI_data.csv')
```

```
print(df.head(5))
```

```
   LYLTY_CARD_NBR        DATE  STORE_NBR  TXN_ID  PROD_NBR  \
0            1000  2018-10-17          1       1         5
1            1002  2018-09-16          1       2        58
2            1003  2019-03-07          1       3        52
3            1003  2019-03-08          1       4       106
4            1004  2018-11-02          1       5        96

                               PROD_NAME  PROD_QTY  TOT_SALES  PACK_SIZE  \
0  Natural Chip        Compny SeaSalt175g         2        6.0        175
1   Red Rock Deli Chikn&Garlic Aioli 150g         1        2.7        150
2   Grain Waves Sour    Cream&Chives 210G         1        3.6        210
3  Natural ChipCo      Hony Soy Chckn175g         1        3.0        175
4          WW Original Stacked Chips 160g         1        1.9        160

        BRAND              LIFESTAGE PREMIUM_CUSTOMER
0     NATURAL  YOUNG SINGLES/COUPLES          Premium
1         RRD  YOUNG SINGLES/COUPLES       Mainstream
2     GRNWVES         YOUNG FAMILIES           Budget
3     NATURAL         YOUNG FAMILIES           Budget
4  WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream
```

checking missing values

```
df.isnull().sum()
```

```
LYLTY_CARD_NBR      0
DATE                0
STORE_NBR           0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
PACK_SIZE           0
BRAND               0
LIFESTAGE           1
PREMIUM_CUSTOMER    1
dtype: int64
```

removing missing values

```
df.dropna(inplace = True)
```

verifying if the missing values are removed

```
df.isnull().sum()
```

```
LYLTY_CARD_NBR      0
DATE                0
STORE_NBR           0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
PACK_SIZE           0
BRAND               0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

```
df.info()
```

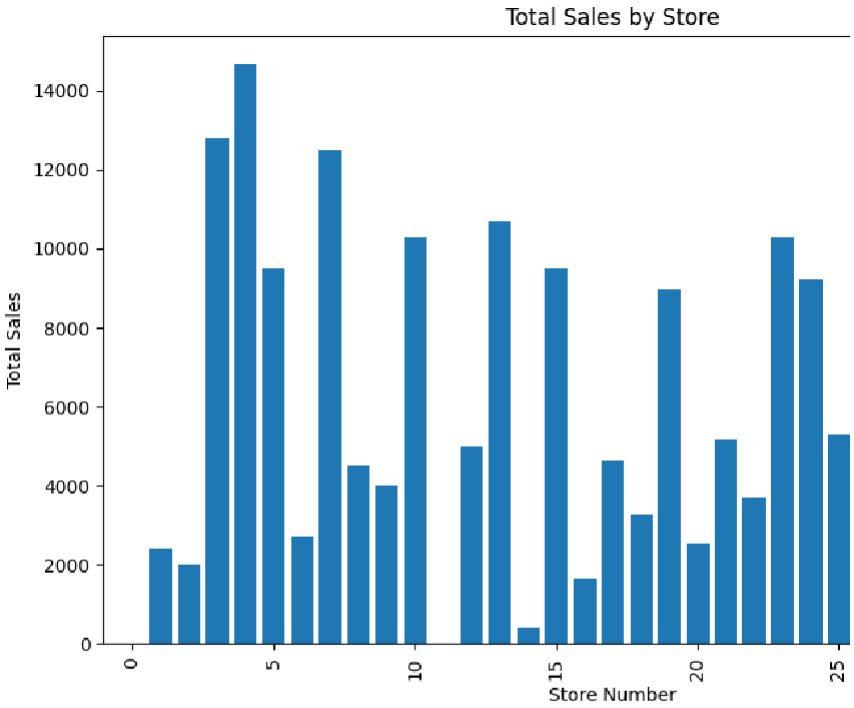
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29432 entries, 0 to 29431
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------------------------------------  --------
 0   LYLTY_CARD_NBR    29432 non-null  int64
 1   DATE              29432 non-null  object
 2   STORE_NBR         29432 non-null  int64
 3   TXN_ID            29432 non-null  int64
 4   PROD_NBR          29432 non-null  int64
```

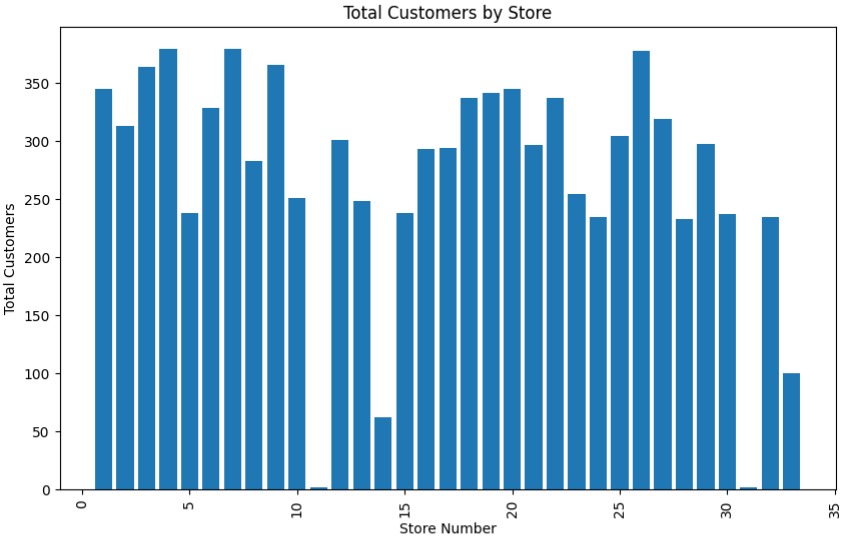
```
 5   PROD_NAME         29432 non-null  object
 6   PROD_QTY          29432 non-null  int64
 7   TOT_SALES         29432 non-null  float64
 8   PACK_SIZE         29432 non-null  int64
 9   BRAND             29432 non-null  object
 10  LIFESTAGE         29432 non-null  object
 11  PREMIUM_CUSTOMER 29432 non-null object
dtypes: float64(1), int64(6), object(5)
memory usage: 2.9+ MB
```

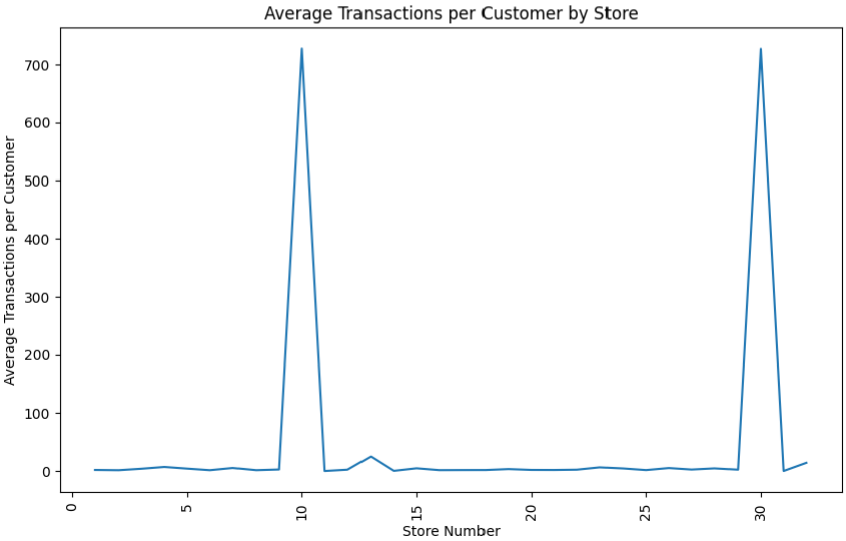
```
# Calculate total sales revenue for each store
total_sales = df.groupby('STORE_NBR')['TOT_SALES'].sum().reset_index()

# Calculate total number of customers for each store
total_customers = df.groupby('STORE_NBR')['LYLTY_CARD_NBR'].nunique().reset_index()

# Calculate average number of transactions per customer for each store
avg_transactions_per_customer = df.groupby('STORE_NBR')['TXN_ID'].nunique() / total_customers['LYLTY_CARD_NBR']
avg_transactions_per_customer = avg_transactions_per_customer.reset_index()
avg_transactions_per_customer.columns = ['STORE_NBR', 'AVG_TRANSACTIONS_PER_CUSTOMER']  # Adding column name

# Combine the results into a single DataFrame
monthly_sales_experience = pd.merge(total_sales, total_customers, on='STORE_NBR')
monthly_sales_experience = pd.merge(monthly_sales_experience, avg_transactions_per_customer, on='STORE_NBR')

# Print the results
print(monthly_sales_experience.head())
```

```
   STORE_NBR  TOT_SALES  LYLTY_CARD_NBR  AVG_TRANSACTIONS_PER_CUSTOMER
0          1    2393.60             345                       1.827476
1          2    2005.80             313                       1.387363
2          3   12802.45             364                       3.928760
3          4   14647.65             379                       7.004202
4          5    9500.80             238                       4.140244
```

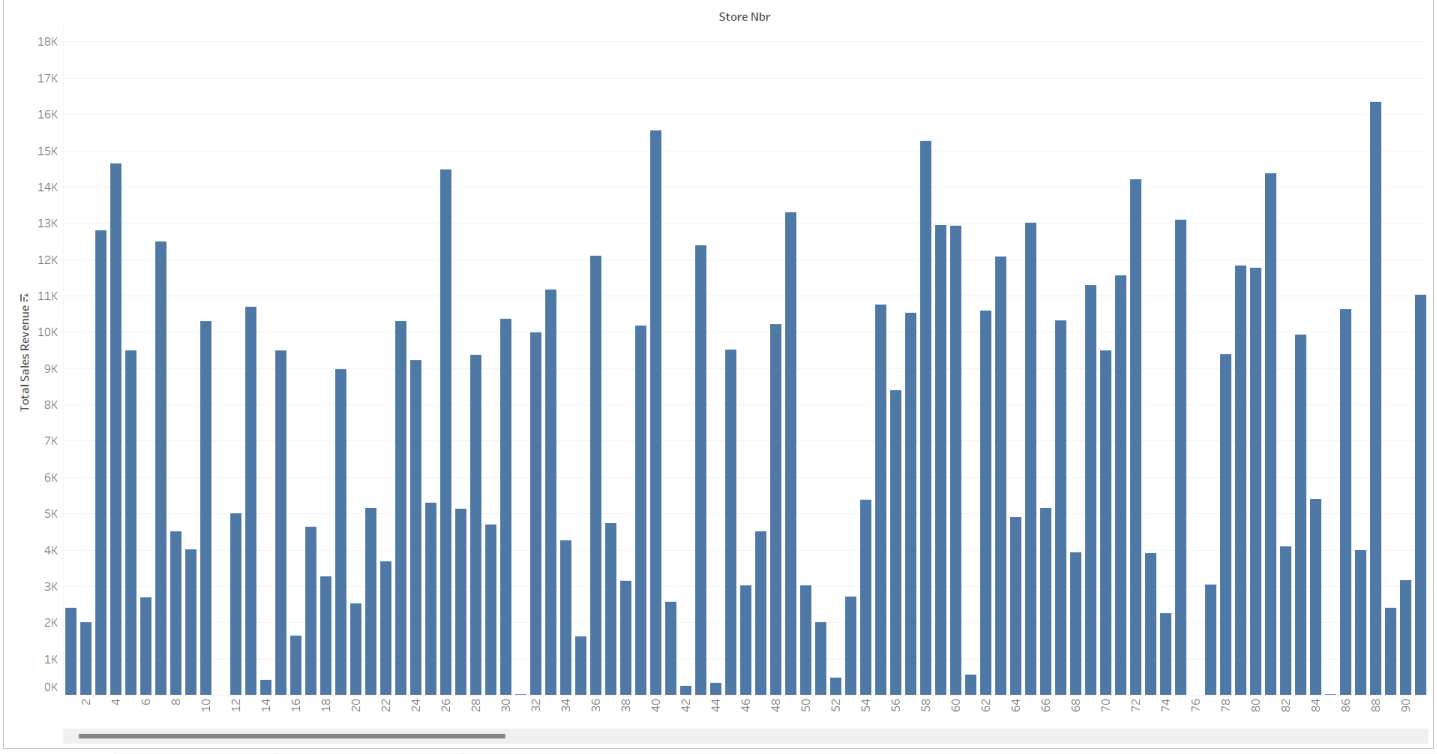
```
import matplotlib.pyplot as plt

# Bar chart for total sales by store
plt.figure(figsize=(10, 6))
plt.bar(monthly_sales_experience['STORE_NBR'], monthly_sales_experience['TOT_SALES'])
plt.xlabel('Store Number')
plt.ylabel('Total Sales')
plt.title('Total Sales by Store')
plt.xticks(rotation=90)
plt.show()

# Bar chart for total customers by store
plt.figure(figsize=(10, 6))
plt.bar(monthly_sales_experience['STORE_NBR'], monthly_sales_experience['LYLTY_CARD_NBR'])
plt.xlabel('Store Number')
plt.ylabel('Total Customers')
plt.title('Total Customers by Store')
plt.xticks(rotation=90)
plt.show()

# Line chart for average transactions per customer by store
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales_experience['STORE_NBR'], monthly_sales_experience['AVG_TRANSACTIONS_PER_CUSTOMER'])
plt.xlabel('Store Number')
plt.ylabel('Average Transactions per Customer')
plt.title('Average Transactions per Customer by Store')
plt.xticks(rotation=90)
plt.show()
```


Total Sales by Store
Total Sales
Store Number
0
2000
4000
6000
8000
10000
12000
14000
0
5
10
15
20
25


Total Customers by Store
Total Customers
Store Number
0
50
100
150
200
250
300
350
0
5
10
15
20
25
30
35

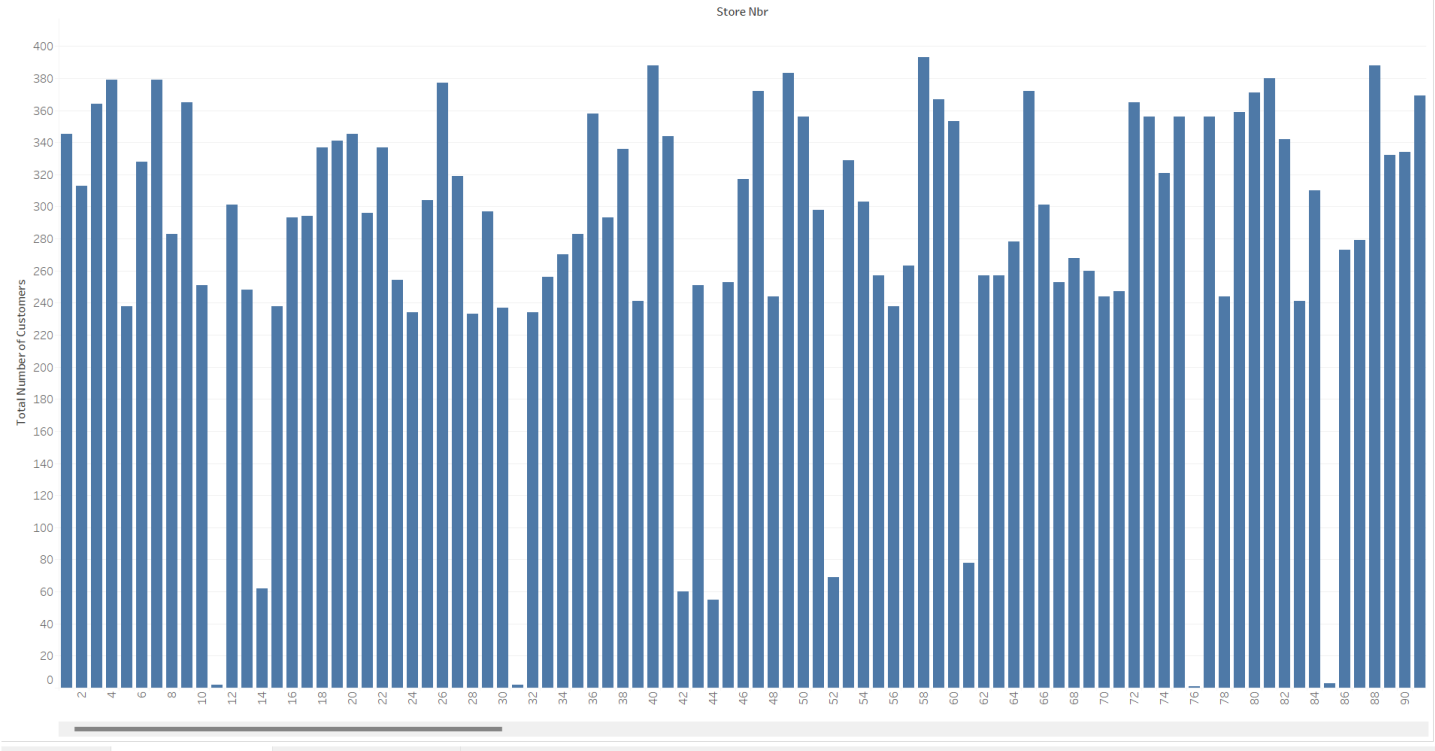

Average Transactions per Customer by Store
Average Transactions per Customer
Store Number
0
100
200
300
400
500
600
700
0
5
10
15
20
25
30

Total Sales Revenue

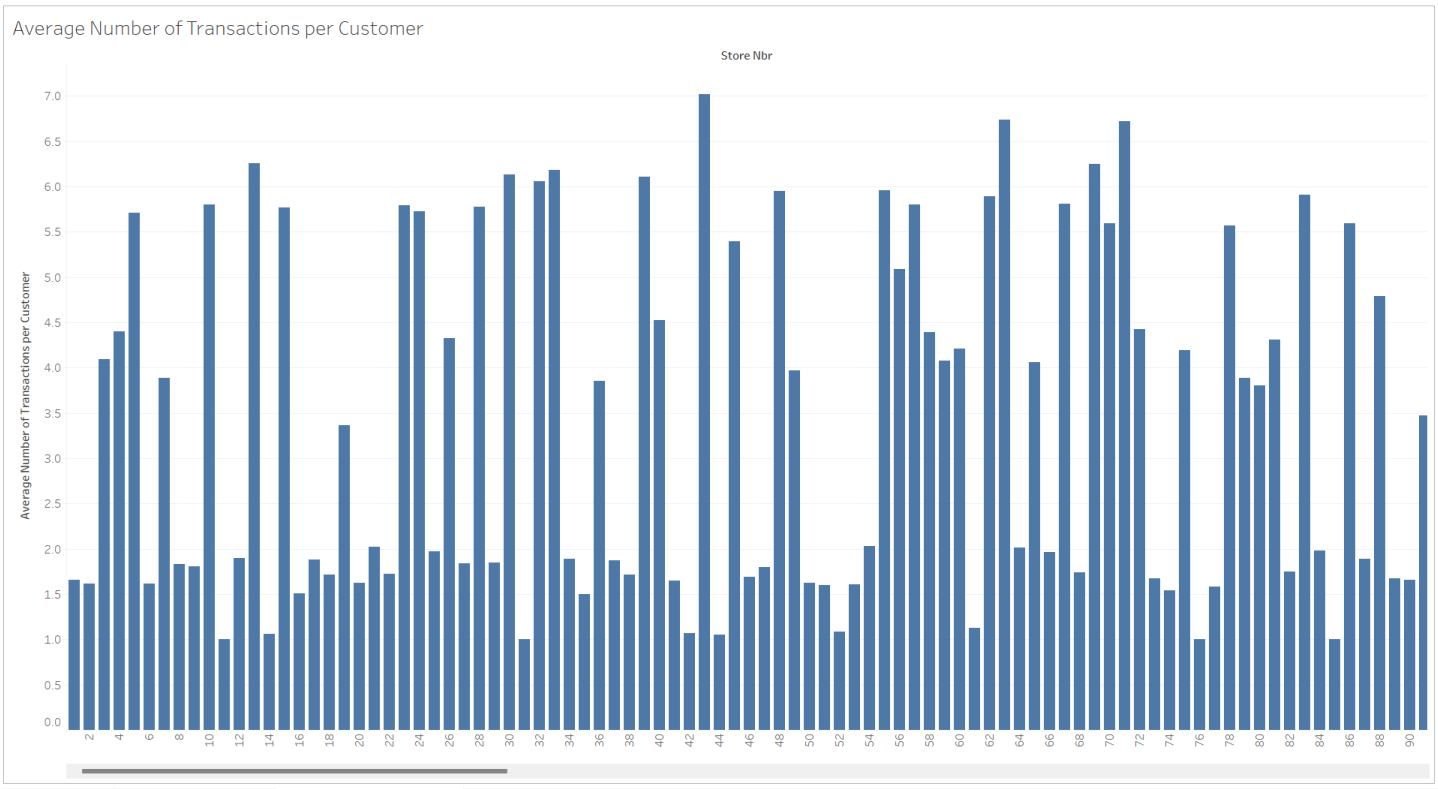


Total sales revenue | Total Number of Customers | Average Number of Transaction...

Total Number of Customers



Total sales revenue | **Total Number of Customers** | Average Number of Transaction...



Total sales revenue | Total Number of Customers | **Average Number of Transactio...**

```
def evaluate_store_trial(trial_store):
    # Filter the DataFrame for the trial store
    trial_store_data = monthly_sales_experience[monthly_sales_experience['STORE_NBR'] == trial_store]

    # Check if there's any data for the trial store
    if trial_store_data.empty:
        print(f"No data found for store {trial_store}.")
        return

    # Calculate the average values for control stores
    control_stores = [store for store in monthly_sales_experience['STORE_NBR'] if store != trial_store]
    control_store_data = monthly_sales_experience[monthly_sales_experience['STORE_NBR'].isin(control_stores)]
    avg_control_sales = control_store_data['TOT_SALES'].mean()
    avg_control_customers = control_store_data['LYLTY_CARD_NBR'].mean()
    avg_control_transactions = control_store_data['AVG_TRANSACTIONS_PER_CUSTOMER'].mean()

    # Check if there's any data for control stores
    if control_store_data.empty:
        print(f"No data found for control stores.")
        return

    # Calculate the percentage change for the trial store compared to control stores
    pct_change_sales = (trial_store_data['TOT_SALES'].iloc[0] - avg_control_sales) / avg_control_sales * 100
    pct_change_customers = (trial_store_data['LYLTY_CARD_NBR'].iloc[0] - avg_control_customers) / avg_control_customers * 100
    pct_change_transactions = (trial_store_data['AVG_TRANSACTIONS_PER_CUSTOMER'].iloc[0] - avg_control_transactions) / avg_control_tran

    # Print the results
    print(f"Store {trial_store} Performance:")
    print(f"Total Sales: {pct_change_sales:.2f}% change from control stores")
    print(f"Total Customers: {pct_change_customers:.2f}% change from control stores")
    print(f"Average Transactions per Customer: {pct_change_transactions:.2f}% change from control stores")

# Evaluate the performance of each trial store
evaluate_store_trial(77)
evaluate_store_trial(86)
evaluate_store_trial(88)
```

```
No data found for store 77.
No data found for store 86.
No data found for store 88.
```

Generate	Create a measure to compare different control stores to each of the trial stores to do this write a function to reduce having to re-do the analysis for each trial store. Consider using Pearson correlations or a metric such as a magnitude distance e.g. 1- (Observed distance – minimum distance)/(Maximum distance – minimum distance) as a measure.	Close

1 of 4 Undo Changes Use code with caution

```
import numpy as np
import pandas as pd

def store_distance(trial_store, control_stores):
    """
    Calculates the distance between a trial store and a list of control stores.

    Args:
      trial_store: The trial store number.
      control_stores: A list of control store numbers.

    Returns:
      A DataFrame containing the distance measures for each control store.
    """

    # Filter the DataFrame for the trial store and control stores
    trial_store_data = monthly_sales_experience[monthly_sales_experience['STORE_NBR'] == trial_store]
    control_store_data = monthly_sales_experience[monthly_sales_experience['STORE_NBR'].isin(control_stores)]

    # Check if trial store data is empty
    if trial_store_data.empty:
        print(f"No data found for trial store {trial_store}.")
        return None

    # Check if control store data is empty
    if control store data.empty:
```

```
        print("No data found for control stores.")
        return None

    # Calculate the distance measures
    distances = []
    for _, trial_row in trial_store_data.iterrows():
        for _, control_row in control_store_data.iterrows():
            # Calculate the Pearson correlation coefficient
            pearson_corr = np.corrcoef(trial_row['TOT_SALES'], control_row['TOT_SALES'])[0, 1]

            # Calculate the magnitude distance
            magnitude_dist = 1 - (abs(trial_row['TOT_SALES'] - control_row['TOT_SALES']) /
                                  (max(trial_row['TOT_SALES'], control_row['TOT_SALES']) -
                                  min(trial_row['TOT_SALES'], control_row['TOT_SALES'])))

            # Store the distance measures
            distances.append({'Trial_Store': trial_store, 'Control_Store': control_row['STORE_NBR'],
                              'Pearson_Correlation': pearson_corr, 'Magnitude_Distance': magnitude_dist})

    # Create a DataFrame from the distance measures
    return pd.DataFrame(distances)


# Example usage:
trial_stores = [77, 86, 88]
control_stores = [store for store in monthly_sales_experience['STORE_NBR'] if store not in trial_stores]

for trial_store in trial_stores:
    print(f"Distances for Trial Store {trial_store}:")
    result = store_distance(trial_store, control_stores)
    if result is not None:
        print(result)
    print("\n")
```

```
Distances for Trial Store 77:
No data found for trial store 77.


Distances for Trial Store 86:
No data found for trial store 86.


Distances for Trial Store 88:
No data found for trial store 88.
```

```
import scipy.stats

# Perform a t-test to compare total sales during the trial period
for trial_store in trial_stores:
    # Filter the data for the trial store and its matched control stores
    trial_store_data = monthly_sales_experience[monthly_sales_experience['STORE_NBR'] == trial_store]
    control_store_data = monthly_sales_experience[monthly_sales_experience['STORE_NBR'].isin(control_stores)]

    # Perform a t-test on total sales
    t_stat, p_value = scipy.stats.ttest_ind(trial_store_data['TOT_SALES'], control_store_data['TOT_SALES'])

    # Print the results
    print(f"Trial Store {trial_store}:")
    print(f"t-statistic: {t_stat}")
    print(f"p-value: {p_value}")

    # Check if the difference is statistically significant
    if p_value < 0.05:
        print("Total sales are significantly different in the trial period.")
    else:
        print("Total sales are not significantly different in the trial period.")

    # If the difference is significant, investigate the drivers of change
    if p_value < 0.05:
        # Compare the number of customers
        t_stat, p_value = scipy.stats.ttest_ind(trial_store_data['LYLTY_CARD_NBR'], control_store_data['LYLTY_CARD_NBR'])
        if p_value < 0.05:
            print("The difference in total sales is driven by a change in the number of customers.")
        else:
            print("The difference in total sales is driven by a change in the average number of transactions per customer.")

    print("\n")
```

```
Trial Store 77:
t-statistic: nan
p-value: nan
Total sales are not significantly different in the trial period.


Trial Store 86:
t-statistic: nan
p-value: nan
Total sales are not significantly different in the trial period.


Trial Store 88:
t-statistic: nan
p-value: nan
Total sales are not significantly different in the trial period.
```